Feature Selection Algorithms: A Survey and Experimental Evaluation

Luis Carlos Molina, Lluís Belanche, Àngela Nebot
Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informátics
Jordi Girona 1-3, C6, 08034, Barcelona, Spain.
{lcmolina,belanche,angela}@lsi.upc.es

Abstract

In view of the substantial number of existing feature selection algorithms, the need arises to count on criteria that enables to adequately decide which algorithm to use in certain situations. This work assesses the performance of several fundamental algorithms found in the literature in a controlled scenario. A scoring measure ranks the algorithms by taking into account the amount of relevance, irrelevance and redundance on sample data sets. This measure computes the degree of matching between the output given by the algorithm and the known optimal solution. Sample size effects are also studied.

1 Introduction

The feature selection problem in terms of supervised inductive learning is: given a set of candidate features select a subset defined by one of three approaches: a) the subset with a specified size that optimizes an evaluation measure, b) the subset of smaller size that satisfies a certain restriction on the evaluation measure and c) the subset with the best commitment among its size and the value of its evaluation measure (general case). The generic purpose pursued is the improvement of the inductive learner, either in terms of learning speed, generalization capacity or simplicity of the representation. It is then possible to understand better the results obtained by the inducer, diminish its volume of storage, reduce the noise generated by irrelevant or redundant features and eliminate useless knowledge.

A feature selection algorithm (FSA) is a computational solution that is motivated by a certain definition of *relevance*. However, the relevance of a feature –as seen from the inductive learning perspective– may have several definitions depending on the objective that is looked for. An irrelevant feature is not useful for induction, but not all relevant features are necessarily useful for induction[5].

In this research, several fundamental algorithms found in the literature are studied to assess their performance in a controlled scenario. To this end, a measure to evaluate FSAs is proposed that takes into account the particularities of relevance, irrelevance and redundance on the sample data set. This measure computes the degree of matching between the output given by a FSA and the known optimal solution. Sample size effects are also studied. The results illustrate the strong dependence on the particular conditions of the FSA used and on the amount of irrelevance and redundance in the data set description, relative to the total number of features. This should prevent the use of a single algorithm specially when there is poor knowledge available about the structure of the solution.

2 Algorithms for Feature Selection

A FSA should be seen as a computational approach to a definition of relevance, although in many cases these definitions are followed in a somewhat loose sense. For a review of such definitions, see [16].

2.1 Feature Selection Definition

Let X be the original set of features, with cardinality $|X| = n$. The *continuous* feature selection problem refers to the assignment of weights w_i to each feature $x_i \in X$ in such a way that the order corresponding to its theoretical relevance is preserved. The *binary* feature selection problem refers to the assignment of binary weights. This can be carried out directly (like many FSAs in machine learning [4, 9]), or *filtering* the output of the continuous problem solution (see §4.1). These are quite different problems reflecting different design objectives. In the continuous case, one is interested in keeping all the features but in using them differentially in the learning process. On the contrary, in the binary case one is interested in keeping just a subset of the features and using them equally in the learning process.

The feature selection problem can be seen as a search in a hypothesis space (set of possible solutions). In the case of the binary problem, the number of potential subsets to evaluate is 2^n. In this case, a general definition is [13]:

0-7695-1754-4/02 $17.00 © 2002 IEEE

Definition 1 (Feature Selection) *Let $J(X')$ be an evaluation measure to be optimized (say to maximize) defined as $J: X' \subseteq X \rightarrow \mathbb{R}$ The selection of a feature subset can be seen under three considerations:*

- *Set $|X'| = m < n$. Find $X' \subset X$, such that $J(X')$ is maximum.*
- *Set a value J_o, this is, the minimum J that is going to be tolerated. Find the $X' \subseteq X$ with smaller $|X'|$, such that $J(X') \geq J_o$.*
- *Find a compromise among minimizing $|X'|$ and maximizing $J(X')$ (general case).*

Notice that, with these definitions, an optimal subset of features is not necessarily unique.

2.2 Characterization of FSAs

There exist in the literature several considerations to characterize feature selection algorithms [3, 8, 14]. In view of them it is possible to describe this characterization as a search problem in the hypothesis space as follows:

- **Search Organization.** General strategy with which the space of hypothesis is explored. This strategy is in relation to the portion of hypothesis explored with respect to their total number.
- **Generation of Successors.** Mechanism by which possible variants (successor candidates) of the current hypothesis are proposed.
- **Evaluation Measure.** Function by which successor candidates are evaluated, allowing to *compare* different hypothesis to guide the search process.

2.2.1 Search Organization

A search algorithm is responsible for driving the feature selection process using a specific strategy. Each *state* in the search space specifies a *weighting* $w_1, \ldots, w_n$ of the possible features of X, with $|X| = n$. In the binary case, $w_i \in \{0, 1\}$, whereas in the continuous case $w_i \in [0, 1]$. Notice we are stating that relevance should be upper and lower bounded. Also in the binary case a partial order $\prec$ exists in the search space, with $S_1 \prec S_2$ if $S_1 \subset S_2$, whereas in the continuous case $S_1 \prec S_2$ if, for all i, $w_i(S_1) \leq w_i(S_2)$ holds. Being L a (labeled) list of weighed subsets of features (i.e. states), L maintains the (ordered) current list of solutions. The labels indicate the value of the evaluation measure. We consider three types of search: exponential, sequential and random. Most sequential algorithms are characterized by $|\mathrm{L}| = 1$, whereas exponential and random ones typically use $|\mathrm{L}| \geq 1$.

Exponential Search: It corresponds to algorithms that carry out searches whose cost is $O(2^n)$. The exhaustive search is an optimal search, in the sense that the best solution is guaranteed. An optimal search need not be exhaustive; for example, if an evaluation measure is *monotonic* a BRANCH AND BOUND[17] algorithm is optimal. A measure J is monotonic if for any two subsets S_1, S_2 and $S_1 \subseteq S_2$, then $J(S_1) \geq J(S_2)$. Another example would be an A^* search with an admissible heuristic[18].

Sequential Search: This sort of search selects one among all the successors to the current state. This is done in an iterative manner and once the state is selected it is not possible to go back. Although there is no explicit backtracking the number of such steps must be limited by $O(n)$ in order to qualify as a sequential search. The complexity is determined taking into account the number k of evaluated subsets in each state change. The cost of this search is therefore polynomial $O(n^{k+1})$. Consequently, these methods do not guarantee an optimal result, since the optimal solution could be in a region of the search space that is not visited.

Random Search: The idea underlying this type of search is to use its randomness to avoid the algorithm to stay on a local minimum and to allow temporarily moving to other states with worse solutions. These are *anytime* algorithms[14] and can give several optimal subsets as solution.

2.2.2 Generation of Successors

All of the operators act by modifying in some way the weights w_i of the features x_i, with $w_i \in \mathbb{R}$ (in the case of the *weighting* operator), or $w_i \in \{0, 1\}$ (for the rest).

Forward: Add features to the current solution X', among those that have not been selected yet. In each step, the feature that makes J be greater is added to the solution. Starting with $X' = \emptyset$, the *forward* step consists of:

$$X' := X' \cup \{x_i \in X \setminus X' \mid J(X' \cup \{x_i\}) \text{ is bigger}\} \quad (1)$$

The stopping criterion can be: $|X'| = n'$ (if n' has been fixed in advance), the value of J has not increased in the last j steps, or it surpasses a prefixed value J_0. The cost of the operator is $O(n)$. The main disadvantage is that it is not possible to have in consideration certain basic interactions among features. For example, if x_1, x_2 are such that $J(\{x_1, x_2\}) \gg J(\{x_1\}), J(\{x_2\})$, neither x_1 and x_2 could be selected, in spite of being very useful.

Backward: Remove features from the current solution X', among those that have not been removed yet. In each step, the feature that makes J be greater is removed from the solution. Starting with $X' = X$, the *backward* step is:

$$X' := X' \setminus \{x_i \in X' \mid J(X' \setminus \{x_i\}) \text{ is bigger}\} \quad (2)$$

The stopping criterion can be: $|X'| = n'$, the value of J has not increased in the last j steps, or it falls below a prefixed value J_0. This operator remedies some problems although there still will be many *hidden* interactions (in the sense of being unobtainable). The cost is $O(n)$, although in practice it demands more computation than *forward* [12].

Both operators (forward and backward) can be generalized selecting, at each step, subsets of k elements X'' and

selecting the one making $J(X' \cup X'')$ or $J(X' \setminus X'')$ bigger, respectively. The cost of the operator is then $O(n^k)$.

Compound: Apply f consecutive forward steps *and* b consecutive backward ones. If $f > b$ the result is a forward operator, otherwise it is a backward one. An interesting approach is to perform the forward *or* the backward steps, depending on the respective values of J. This allows to discover new interactions among features. An interesting "backtracking mechanism" is obtained, although other stopping conditions should be established if $f = b$. For example, for $f = b = 1$, if x_i is added and x_j is removed, this could be undone in the following steps. A possible stopping criterion is $x_i = x_j$. In sequential FSA, $f \neq b$ assures a maximum of n steps, with a cost $O(n^{f+b+1})$.

Weighting: In the weighting operators, the search space is continuous, and all of the features are present in the solution to a certain degree. A successor state is a state with a different weighting. This is typically done by iteratively sampling the available set of instances.

Random: This group includes those operators that can potentially generate *any other* state in a single step. The rest of operators can also have random components, but they are restricted to some criterion of "advance" in the number of features or in improving the measure J at each step.

2.2.3 Evaluation Measures

Probability of error: Provided the ultimate goal is to build a classifier able of correctly labelling instances generated by the same probability distribution, minimizing the (bayesian) probability of error P_e of the classifier seems to be the most natural choice. Therefore, it is also a clear choice for J.

Let $\vec{x} \in \mathbb{R}^n$ represent the unlabeled instances, and $\Omega = \{\omega_1, \ldots, \omega_m\}$ a set of labels (classes), so that $c : \mathbb{R}^n \to \Omega$. Such probability is defined as [7]:

$$P_e = \int [1 - \max_i P(\omega_i|\vec{x})] p(\vec{x}) d\vec{x} \tag{3}$$

where $p(\vec{x}) = \sum_{i=1}^{m} p(\vec{x}|\omega_i) P(\omega_i)$ is the (unconditional) probability distribution of the instances, and $P(\omega_i|\vec{x})$ is the *a posteriori* probability of ω_i being the class of $\vec{x}$.

Since the class-conditional densities are usually unknown, they can either be explicitly modeled (using parametric or non-parametric methods) or implicitly via the design of a classifier that builds the respective decision boundaries between the classes [7]. Some of these classifiers, like the one-nearest-neighbor rule, have a direct relation to the probability of error. This may require the use of more elaborate methods than a simple holdout procedure (cross validation, bootstrapping) in order to yield a more reliable value.

Divergence: These measures compute a probabilistic distance or *divergence* among the class-conditional probability densities $p(\vec{x}|\omega_i)$, using the general formula:

$$J = \int f[p(\vec{x}|\omega_1), p(\vec{x}|\omega_2)] d\vec{x} \tag{4}$$

To qualify as a valid measure, the function f must be such that the value of J satisfies the following conditions: (a) $J \geq 0$, (b) $J = 0$ only when the $p(\vec{x}|\omega_i)$ are equal and (c) J is maximum when they are non-overlapping. If the features used in a solution $X' \subset X$ are good ones, the divergence among the conditional probabilities will be significant. Poor features will result in very similar probabilities. Some classical choices are: Chernoff, Bhattacharyya, Kullback–Liebler, Kolmogorov and Matusita [7].

Dependence: These measures quantify how strongly two features are associated with one another, in the sense that knowing the value of one it is possible to predict the value of the other. In the context of feature selection, a feature is better evaluated the better it predicts the class.

Interclass distance: These measures are based on the assumption that instances of a different class are distant in the instance space. It is enough then to define a metric between classes and use it as measure:

$$D(\omega_i, \omega_j) = \frac{1}{N_i N_j} \sum_{k_1}^{N_i} \sum_{k_2=k_1+1}^{N_j} d(x_{(i,k_1)}, x_{(j,k_2)}) \tag{5}$$

$$J = \sum_{i=1}^{m} P(\omega_i) \sum_{j=i+1}^{m} P(\omega_j) D(\omega_i, \omega_j) \tag{6}$$

being $x_{(i,j)}$ the instance j of class ω_i, and N_i the number of instances of the class ω_i. The most usual distances d belong to the Euclidean family. These measures do not require the modeling of any density function, but their relation to the probability of error can be very loose.

Information or Uncertainty: Similarly to the probabilistic dependence, we may observe $\vec{x}$ and compute the a posteriori probabilities $P(\omega_i|\vec{x})$ to determine how much information on the class of $\vec{x}$ has been gained, with respect to its prior probability. If all the classes become roughly equally probable, then the information gain is minimal and the uncertainty (entropy) is maximum.

Consistency: An *inconsistency* in X' and S is defined as two instances in S that are equal when considering only the features in X' and that belong to different classes. The aim is thus to find the *minimum* subset of features leading to zero inconsistencies [1]. The *inconsistency count* of an instance $A \in S$ is defined as [14]:

$$IC_{X'}(A) = X'(A) - \max_k X'_k(A) \tag{7}$$

where $X'(A)$ is the number of instances in S equal to A using only the features in X' and $X'_k(A)$ is the number of instances in S of class k equal to A using only the features in X'. The *inconsistency rate* of a feature subset in a sample S is then:

$$IR(X') = \frac{\sum_{A \in S} IC_{X'}(A)}{|S|} \tag{8}$$

This is a monotonic measure, in the sense

$$X_1 \subset X_2 \Rightarrow IR(X_1) \geq IR(X_2).$$

2.3 General Schemes for Feature Selection

The relationship between a FSA and the inducer chosen to evaluate the usefulness of the feature selection process can take three main forms: *embedded*, *filter* and *wrapper*.

Embedded Scheme: The inducer has its own FSA (either explicit or implicit). The methods to induce logical conjunctions[20] provide an example of this embedding. Other traditional machine learning tools like decision trees or artificial neural networks are included in this scheme[15].

Filter Scheme: If the feature selection process takes place before the induction step, the former can be seen as a filter of non-useful features prior to induction. In a general sense it can be seen as a particular case of the embedded scheme in which feature selection is used as a pre-processing. The filter schemes are independent of the induction algorithm.

Wrapper Scheme: In this scheme the relationship is taken the other way around: it is the FSA that uses the learning algorithm as a subroutine[11]. The general argument in favor of this scheme is to equal the bias of both the FSA and the learning algorithm that will be used later on to assess the goodness of the solution. The main disadvantage is the computational burden that comes from calling the induction algorithm to evaluate each subset of considered features.

2.4 General Algorithm for Feature Selection

An abstract algorithm that unifies the behavior of any FSA is depicted in Fig. 1. In particular, being L a (weighed) list of weighed subsets of features (i.e. states), L keeps the ordered set of solutions in course. Exponential algorithms are typically characterized by $|L| \geq 1$ (examples are BRANCH AND BOUND [17] or A^* [18]). The presence in the list is a function of the evaluation measure and defines the expansion order. Heuristic search algorithms also keep this list (of open nodes), and the weighting is the value of the heuristic. Random search methods as Evolutionary Algorithms [2] are characterized by $|L| \geq 1$ (the list is the population and the weighting is the fitness value of the individuals). Sequential algorithms maintain $|L| = 1$, though there are exceptions (e.g., a bidirectional algorithm [8] would use $|L| = 2$). The second weighting (on the features of each solution subset) allows to include the two types of FSA according to their outcome (see §2.1).

The initial list L is in general built out of the original set of features and the algorithm maintains the best solution at all times (*Solution*). At each step, a FSA with a given search organization manipulates the list in a specific way and calls its mechanism for the generation of successors which in turn uses J. The result is an updated list and the eventual update of the best solution found so far. Notice that the data sample S is considered global to the algorithm.

3 Empirical Evaluation of FSAs

The first question arising in relation to a feature selection experimental design is: what are the aspects that we would like to evaluate of a FSA solution in a given data set? In this study we decided to evaluate FSA performance with respect to four particularities: relevance, irrelevance, redundance and sample size. To this end, several fundamental FSAs are studied to assess their performance on synthetic data sets with known relevant features. Then sample data sets of different sizes are corrupted with irrelevant and/or redundant features. The experiments are designed to test the *endurance* of different FSAs (e.g., behaviour against the ratio number-of-irrelevant vs. number-of-relevant features).

3.1 Particularities to be evaluated

Relevance: Different families of problems are generated by varying the number of relevant features N_R. These are features that, by construction, have an influence on the output and whose role can not be assumed by the rest (i.e., there is no redundance).

Irrelevance: Irrelevant features are defined as those features not having any influence on the output, and whose values are generated at random for each example. For a problem *with* N_R *relevant features, different numbers of irrelevant* features N_I are added to the corresponding data sets (thus providing with several subproblems for each choice of N_R).

Redundance: In these experiments, a redundance exists whenever a feature can take the role of another (perhaps the simplest way to model redundance). This is obtained by choosing a relevant feature randomly and replicating it in the data set. For a problem with N_R relevant features, different numbers of redundant features $N_{R'}$ are added in a way analogous to the generation of irrelevant features.

```
Input:
   S - data sample with features X, |X| = n
   J - evaluation measure to be maximized
   GS - successor generation operator
Output:
   Solution - (weighed) feature subset

L := Start_Point(X);
Solution := {best of L according to J};
repeat
   L := Search_Strategy(L, GS(J), X);
   X' := {best of L according to J};
   if J(X') ≥ J(Solution) or (J(X') = J(Solution)
        and |X'| < |Solution|)
   then Solution := X';
until Stop(J, L)
```

Fig. 1. General Feature Selection Algorithm.

Sample Size: It refers to the number of instances $|S|$ of a data sample S. In these experiments, $|S|$ is defined as $|S| = \alpha k N_T c$, where α is a constant, k is a multiplying factor, N_T is the total number of features $(N_R + N_I + N_{R'})$ and c is the number of classes. Note this means that the sample size will depend linearly on the number of features.

3.2 Evaluation of Performance

The *score* criterion expresses the degree to which a solution obtained by a FSA matches the correct solution. This criterion behaves as a similarity $s(x, y) : X \times X \to [0, 1]$ in the classical sense [6], satisfying:

1. $s(x, y) = 1 \iff x = y$
2. $s(x, y) = s(y, x)$

where $s(x, y) > s(x, z)$ indicates that y is more similar to x than z. Let us denote by X the total set of features, partitioned in $X = X_R \cup X_I \cup X_{R'}$, being $X_R, X_I, X_{R'}$ the subsets of relevant, irrelevant and redundant features of X, respectively and call $X^* \subseteq X$ the ideal solution. Let us denote by $\mathcal{A}$ the feature subset selected by a FSA. The idea is to check how much $\mathcal{A}$ and X^* have in common. Let us define $\mathcal{A}_R = X_R \cap \mathcal{A}$, $\mathcal{A}_I = X_I \cap \mathcal{A}$ and $\mathcal{A}_{R'} = X_{R'} \cap \mathcal{A}$. In general, we have $\mathcal{A}_T = X_T \cap \mathcal{A}$ (hereafter T stands for a subindex in $\{R, I, R'\}$). Since necessarily $\mathcal{A} \subseteq X$, we have $\mathcal{A} = \mathcal{A}_R \cup \mathcal{A}_I \cup \mathcal{A}_{R'}$. The *score* $S_X(\mathcal{A}) : P(X) \to [0, 1]$ will fulfill the following conditions:

- $S_X(\mathcal{A}) = 0 \iff \mathcal{A} = X_I$
- $S_X(\mathcal{A}) = 1 \iff \mathcal{A} = X^*$
- $S_X(\mathcal{A}) > S_X(\mathcal{A}')$ indicates that $\mathcal{A}$ is more similar to X^* than $\mathcal{A}'$.

The score is defined in terms of the similarity in that for all $\mathcal{A} \subseteq X, S_X(\mathcal{A}) = s(\mathcal{A}, X^*)$. This scoring measure will also be parameterized, so that it can ponder each type of divergence (in relevance, irrelevance and redundance) to the optimal solution. The set of parameters is expressed as $\alpha = \{\alpha_R, \alpha_I, \alpha_{R'}\}$ with $\alpha_T \geq 0$ and $\sum \alpha_T = 1$.

Intuitive Description The criterion $S_X(\mathcal{A})$ penalizes three situations:

1. There are relevant features lacking in $\mathcal{A}$ (the solution is incomplete).
2. There are more than enough relevant features in $\mathcal{A}$ (the solution is redundant).
3. There are some irrelevant features in $\mathcal{A}$ (the solution is incorrect).

An order of importance and a weight will be assigned (via the α_T parameters), to each of these situations.

Formal Description

The precedent point (3.) is simple to model: if suffices to check whether $|\mathcal{A}_I| > 0$, being $\mathcal{A}$ the solution of the FSA. Relevance and redundance are strongly related given that, in this context, a feature is redundant or not depending on what other relevant features are present in $\mathcal{A}$.

Notice then that the optimal solution X^* is not unique, though all them should be equally valid for the *score*. To this end, the features are broken down in *equivalence classes*, where elements of the same class are redundant to each other (i.e., any optimal solution must comprise only one feature of each equivalence class).

Being $\mathcal{A}$ a feature set, we define a binary relation between two features $x_i, x_j \in \mathcal{A}$ as: $x_i \sim x_j \iff x_i$ and x_j represent the same information. Clearly $\sim$ is an equivalence relation. Let $\mathcal{A}^\sim$ be the quotient set of $\mathcal{A}$ under $\sim$, $\mathcal{A}^\sim = \{[x] \mid x \in \mathcal{A}\}$, any optimal solution $\mathcal{A}^*$ will satisfy:

1. $|\mathcal{A}^*| = |X_R|$
2. $\forall [x_i] \in \mathcal{A}^\sim : \exists x_j \in [x_i] : x_j \in \mathcal{A}^*$

We denote by $\mathcal{A}^*$ *any* of these solutions.

Construction of the *score*

In the present case, the set to be split in equivalence classes is formed by all the relevant features (redundant or not) chosen by a FSA. We define then:

$$\mathcal{A}_R^\sim = (\mathcal{A}_R \cup \mathcal{A}_{R'})^\sim$$

(equivalence classes in which the relevant features chosen by a FSA are split)

$$X_R^\sim = (X_R \cup X_{R'})^\sim$$

(equivalence classes in which the original features are split)

Let $\mathcal{A}_R^\sim \uplus X_R^\sim = \{[x_i] \in X_R^\sim \mid \exists [x_j] \in \mathcal{A}_R^\sim : [x_j] \subseteq [x_i]\}$ and define, for Q quotient set:

$$F(Q) = \sum_{[x] \in Q} (|x| - 1)$$

The idea is to express the *quotient* between the number of redundant features chosen by the FSA and the number it could have chosen, given the relevant features present in its solution. In the precedent notation, this is written (provided the denominator is not null):

$$\frac{F(\mathcal{A}_R^\sim)}{F(\mathcal{A}_R^\sim \uplus X_R^\sim)}$$

Let us finally build the *score*, formed by three terms: relevance, irrelevance and redundance. Defining:

$$I = 1 - \frac{|\mathcal{A}_I|}{|X_I|}, \qquad R = \frac{|\mathcal{A}_R^\sim|}{|X_R|}, \text{ with } \mathcal{A}_R^\sim = (\mathcal{A}_R \cup \mathcal{A}_{R'})^\sim$$

$$R' = \begin{cases} 0 & \text{if } F(\mathcal{A}_R^\sim \uplus X_R^\sim) = 0 \\ \frac{1}{|X_{R'}|}\left(1 - \frac{F(\mathcal{A}_R^\sim)}{F(\mathcal{A}_R^\sim \uplus X_R^\sim)}\right) & \text{otherwise.} \end{cases}$$

the score is $S_X(\mathcal{A}) = \alpha_R R + \alpha_{R'} R' + \alpha_I I, \mathcal{A} \subseteq X$.

Restrictions on the α_T

We can establish now the desired restrictions on the behavior of the score. From the more to the less severe: there are relevant features lacking, there are irrelevant features, and there is redundancy in the solution. This is reflected in the following conditions on the α_T:

1. Choosing an irrelevant feature is better than missing a relevant one: $\frac{\alpha_R}{|X_R|} > \frac{\alpha_I}{|X_I|}$

2. Choosing a redundant feature is better than choosing an irrelevant one: $\frac{\alpha_I}{|X_I|} > \frac{\alpha_{R'}}{|X_{R'}|}$

We also define $\alpha_T = 0$ if $|X_T| = 0$. Notice that the denominators are important for, as an example, expressing the fact that it is not the same choosing an irrelevant feature when there were only two that when there were three (in the latter case, there is an irrelevant feature that could have been chosen when it was not).

4 Experimental Evaluation

In this section we detail the experimental methodology and quantify the various parameters of the experiments. The basic idea consists on generating sample data sets with known particularities (synthetic functions f) and hand them over to the different FSAs to obtained a hypothesis H. The divergence between the defined function and the obtained hypothesis will be evaluated by the *score* criterion. This experimental design is illustrated in Fig. 2.

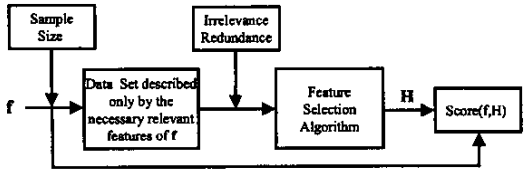


Figure 2. FlowChart of Experimental Design.

4.1 Description of the FSAs used

The ten FSAs used in the experiments were : E-SFG, QBB, LVF, LVI, C-SBG, RELIEF, SFBG, SFFG, W-SBG, and W-SFG (see Table 1). The algorithms E-SFG, W-SFG are versions of SFG using entropy and the accuracy of a C4.5 inducer, respectively. The algorithms C-SBG, W-SBG are versions of SBG using consistency and the accuracy of a C4.5 inducer, respectively. During the course of the experiments the algorithms FOCUS, B&B, ABB and LVW were put aside due to their unaffordable consumption of resources. For a review of all these algorithms, see [16].

Since RELIEF and E-SFG give as output an ordered list of features x_i according to their weight w_i, a filtering criterion is necessary to transform this solution to a subset of features. The procedure used here is simple: since the interest is in determining a good cut point, first those w_i further than two variances from the mean are discarded (that is to say, with very high or very low weights). Then define $s_i = w_i + w_{i-1}$ and $\sigma_j = \sum_{i=2}^{j} s_i$. The objective is to search for the feature x_j such that:

$$1 - \frac{\sigma_j}{\sigma_n}\frac{n-j}{n} \quad \text{is maximum.}$$

The cut point is then set between x_j and x_{j+1}.

Algorithm	Search Organization	Generation of Succesors	Evaluation Measure
LVF	Random	Random	Consistency
LVI	Random	Random	Consistency
QBB	Random/Expon.	Random/Backward	Consistency
RELIEF	Random	Weighting	Distance
C-SBG	Sequential	Backward	Consistency
E-SFG	Sequential	Forward	Entropy
SFBG	Exponential	Compound	Consistency
SFFG	Exponential	Compound	Consistency
W-SBG	Sequential	Backward	Accuracy(C4.5)
W-SFG	Sequential	Forward	Accuracy(C4.5)

Table 1. FSAs used in the experiments.

4.2 Implementations of Data Families

A total of twelve families of data sets were generated studying three different problems and four instances of each, by varying the number of relevant features N_R. Let $x_1, \ldots, x_n$ be the relevant features of a problem f.

Parity: This is the classic binary problem of parity n, where the output is $f(x_1, \cdots, x_n) = 1$ if the number of $x_i = 1$ is odd and $f(x_1, \cdots, x_n) = 0$ otherwise.

Disjunction: A disjunctive task, with $f(x_1, \cdots, x_n) = 1$ if $(x_1 \wedge \cdots \wedge x_{n'}) \vee (x_{n'+1} \wedge \cdots \wedge x_n)$, with $n' = n\,div\,2$ if n is even and $n' = (n\,div\,2) + 1$ if n is odd.

GMonks: This problem is a generalization of the classic *monks* problems [19]. In its original version, three independent problems were applied on sets of $n = 6$ features that take values of a discrete, finite and unordered set (nominal features). Here we have grouped the three problems in a single one computed on *each* segment of 6 features. Let n be multiple of 6, $k = n\,div\,6$ and $b = 6(k'-1)+1$, for $1 \leq k' \leq k$. Let us denote for "1" the first value of a feature, for "2" the second, etc. The problems are the following:

1. $P1 : (x_b = x_{b+1}) \vee x_{b+4} = 1$
2. $P2$: two or more $x_i = 1$ in $x_b \cdots x_{b+5}$
3. $P3 : (x_{b+4} = 3 \wedge x_{b+3} = 1) \vee (x_{b+4} \neq 3 \wedge x_{b+1} \neq 2)$

For each segment, the boolean condition $P2 \wedge \neg(P1 \wedge P3)$ is checked. If this condition is satisfied for s or more segments with $s = n_s\,div\,2$ (being n_s the number of segments) the function *GMonks* is 1; otherwise, it is 0.

4.3 Experimental Setup

The experiments were divided in three groups. The first group refers to the relationship between irrelevance vs. relevance. The second refers to the relationship between redundance vs. relevance. The last group refers to sample size. Each group uses three families of problems (*Parity*, *Disjunction* and *GMonks*) with four different instances for each problem, varying the number of relevant features N_R.

Relevance: The different numbers N_R vary for each problem, as follows: {4, 8, 16, 32} (for *Parity*), {5, 10, 15, 20} (for *Disjunction*) and {6, 12, 18, 24} (for *GMonks*).

Irrelevance: In these experiments, we have N_I running from 0 to 2 times the value of N_R, in intervals of 0.2 (that is, eleven different experiments of irrelevance for each N_R).

Redundance: Similarly to the generation of irrelevant features, we have $N_{R'}$ running from 0 to 2 times the value of N_R, in intervals of 0.2.

Sample Size: Given the formula $|S| = \alpha k N_T c$ (see §3.1), different problems were generated considering $k \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.75, 2.0\}$, $N_T = N_R + N_I + N_{R'}$, $c = 2$, $\alpha = 20$ and $N_I = N_{R'} = N_R \, div \, 2$.

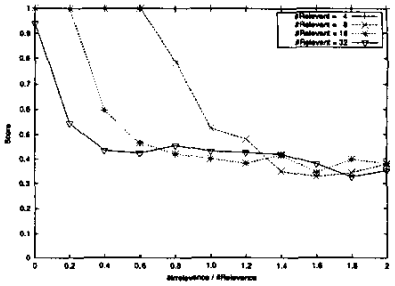
4.4 Results

Due to space reasons, only a sample of the results are presented, in Figs. 3(a) and (b). Each point is the average of 10 independent runs with different random data samples. The horizontal axis represents the ratios between these particularities as explained above. The vertical axis represents the average results given by the score criterion.

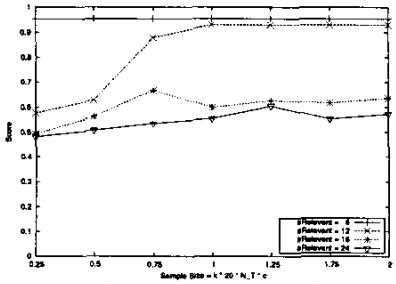
In Fig. 3(a) the C-SBG algorithm shows at first a good performance but clearly as the irrelevance ratio increases, it falls dramatically (below the 0.5 level from $N_I = N_R$ on). Note that for $N_R = 4$ performance is always perfect (the plot is on top of the graphic). The plots in Fig. 3(b) show additional interesting results because we can appreciate the curse of dimensionality effect [10]. In this figure, W-SBG present an increasingly poor performance (see the figure from top to bottom) with the number of features provided the number of examples is increasing in a linear way. However, in general, as long as more examples are added performance is better (see the figure from left to right).

A summary of the results is displayed in Fig. 4 for the ten algorithms, allowing for a comparison across all the sample data sets with respect to each studied particularity. Specifically, Figs. 4(a), (b) and (c) show the average score of each algorithm for irrelevance, redundance and sample size, respectively. In each graphic there are two keys: the key to the left shows the algorithms ordered by *total* average performance, from top to bottom. The key to the right shows the algorithms ordered by average performance on the *last* abscissa value, also from top to bottom. In other words, the left list is topped by the algorithm that wins on average, while the right list is topped by the algorithm that ends on the lead. This is also useful to help reading the graphics.

Fig. 4(a) shows that RELIEF ends up on the lead of the irrelevance vs. relevance problems, while SFFG shows the best average performance. The algorithm W-SFG is also well positioned. Fig. 4(b) shows that the algorithms LVF and LVI together with C-SBG are the overall best. In fact, there is a bunch of algorithms that also includes the two *floating* and QBB showing a close performance. Note how RELIEF and the *wrappers* are very poor performers. Fig. 4(c) shows that the wrapper algorithms seem to be able to extract the most of the data when there is a shortage of it. Surprisingly,



(a) Irrelevance vs. Relevance - Parity - C-SBG.



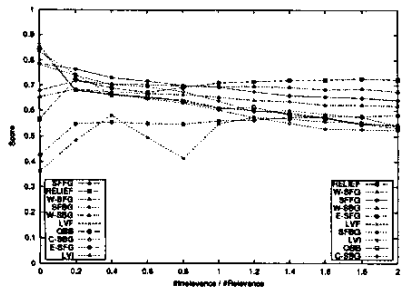
(b) Sample Size - Parity - W-SBG.

Figure 3. Some results to irrelevance, relevance and sample size.

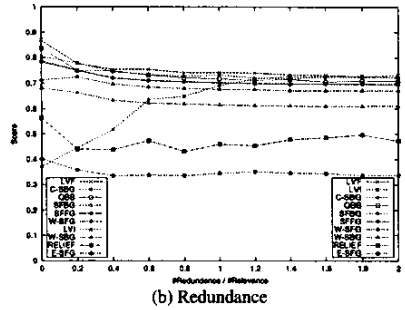
the backward wrapper is just fairly positioned on average. The SFFG is again quite good on average, together with C-SBG. However, all of the algorithms are quite close and show the same kind of dependency to the data. Note the general poor performance of E-SFG, provided it is the only algorithm that computes its evaluation measure (entropy in this case) independently for each feature.

5 Conclusions

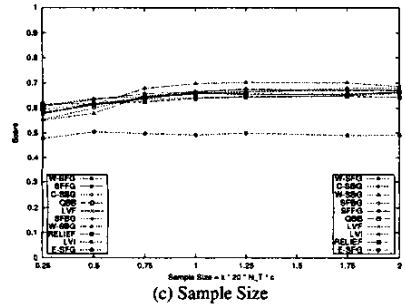
The task of a feature selection algorithm (FSA) is to provide with a computational solution to the feature selection problem motivated by a certain definition of *relevance*. This algorithm should be reliable and efficient. The many FSAs proposed in the literature are based on quite different principles (as the evaluation measure used, the precise way to explore the search space, etc) and loosely follow different definitions of relevance. In this work a way to evaluate FSAs was proposed in order to understand their general behaviour on the particularities of relevance, irrelevance, redundancy and sample size of synthetic data sets. To achieve this goal, a set of controlled experiments using artificially generated data sets were designed and carried out. The set of optimal solutions is then compared with the output given by the FSAs (the obtained hypotheses). To this end, a *scoring* measure was defined to express the degree of approximation of



(a) Irrelevance



(b) Redundance



(c) Sample Size

Figure 4. Results ordered by total average performance on the data sets (left inset) and by end performance (right inset).

the FSA solution to the real solution. The final outcome of the experiments can be seen as an illustrative step towards gaining useful knowledge that enables to decide which algorithm to use in certain situations. The behaviour of the algorithms to different data particularities is shown and thus the danger in relying in a single algorithm. This points in the direction of using new hybrid algorithms or combinations thereof for a more reliable assessment of feature relevance.

Acknowledgements

This work is supported by the Spanish CICyT Project TAP99-0747 and by the Mexican Petroleum Institute.

References

[1] H. Almuallim and T. G. Dietterich. Learning Boolean Concepts in the Presence of Many Irrelevant Features. *Artificial Intelligence*, 69(1–2):279–305, 1994.

[2] T. Back. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, Oxford, 1996.

[3] A. L. Blum and P. Langley. Selection of Relevant Features and Examples in Machine Learning. In R. Greiner and D. Subramanian, editors, *Artificial Intelligence on Relevance*, volume 97, pp. 245–271. AI, 1997.

[4] R. A. Caruana and D. Freitag. Greedy Attribute Selection. In *Proc. of the 11th Int. Conf. on Machine Learning*, pp. 28–36, New Brunswick, NJ, 1994. Morgan Kaufmann.

[5] R. A. Caruana and D. Freitag. How Useful is Relevance? Technical report, AAAI Symposium on Relevance, 1994.

[6] S. Chandon and L. Pinson. *Analyse Typologique*. Masson, 1981.

[7] P. A. Devijver and J. Kittler. *Pattern Recognition – A Statistical Approach*. Prentice Hall, London, GB, 1982.

[8] J. Doak. An Evaluation of Feature Selection Methods and their Application to Computer Security. Technical Report CSE–92–18, Univ. of California, 1992.

[9] M. A. Hall. *Correlation–based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, 1999.

[10] A. K. Jain and D. Zongker. Feature Selection: Evaluation, Application, and Small Sample Performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[11] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant Features and the Subset Selection Problem. In *Proc. of the 11th Int. Conf. on Machine Learning*, pp. 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann.

[12] D. Koller and M. Sahami. Toward Optimal Feature Selection. In *Proc. of the 13th Int. Conf. on Machine Learning*, pp. 284–292, Bari, IT, 1996. Morgan Kaufmann.

[13] M. Kudo and J. Sklansky. A Comparative Evaluation of medium and large–scale Feature Selectors for Pattern Classifiers. In *Proc. of the 1st Int. Workshop on Statistical Techniques in Pattern Recognition*, pp. 91–96, Prague, 1997.

[14] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, London, GB, 1998.

[15] T. M. Mitchell. Generalization as Search. *Artificial Intelligence*, 18(2):203–226, 1982.

[16] L. C. Molina, L. Belanche, and A. Nebot. Feature Selection Algorithms: A Survey and Experimental Evaluation. Technical Report LSI-02-62-R, Universitat Politècnica de Catalunya, Barcelona, Spain, 2002.

[17] P. Narendra and K. Fukunaga. A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on Computer*, C–26(9):917–922, 1977.

[18] J. Pearl. *Heuristics*. Addison Wesley, 1983.

[19] S. B. Thrun et al. The MONK's Problems: A Performance Comparison of Different Learning Algorithms. Technical Report CS-91-197, CMU, Pittsburgh, PA, 1991.

[20] P. H. Winston. Learning Structural Descriptions from Examples. In Winston, P. H., editor, *The Psychology of Computer Vision*, New York, NY, 1975. McGraw Hill.